TP10

SCRABBLE

Dictionnaires

Dans une liste, les éléments sont ordonnés et ont une position bien définie.

Un **dictionnaire** va aussi permettre de rassembler des éléments mais ceux-ci seront identifiés par une **clé**. On peut bien sûr faire l'analogie avec un dictionnaire classique où on accède à une définition avec un mot.

	Syntaxe python
Dictionnaire	`dict = {cle1 : valeur1, cle2 : valeur2, ...}`
Liste des clés	`dict.keys()`
Liste des (clé, valeur)	`dict.items()`
Accéder à la valeur d'une clé	`dict[cle]`
Nombre de clés	`len(dict)`

Exemple : On crée un dictionnaire donnant les résultats d'un sondage.

```
>>> sondage
{'oui': 10, 'non': 8, 'blanc': 3}

>>> sondage.keys()
dict_keys(['oui', 'non', 'blanc'])

>>>sondage.items()
dict_items([('oui', 10), ('non', 8), ('blanc', 3)])

>>> sondage['oui']
10
>>> sondage['oui'] = sondage['oui'] + 1    # on modifie la valeur
>>> sondage['oui']
11
>>> len(sondage)
    3
# Parcourir un dictionnaire
>>> for cle in sondage : # seul le parcours des clés a lieu
...     print("le nombre de", cle, "est", sondage[cle])

le nombre de oui est 11
le nombre de non est 8
le nombre de blanc est 3

>>> "oui" in sondage # les éléments de sondage sont ses clés
    True

>>> 8 in sondage
    False
```

Pour ce TP, vous disposez d'un fichier pré-complété sur le réseau pédagogique :
Classe PCSI → Documents en consultation → Informatique → `TP10-a-completer.py`
Copiez-collez ce fichier dans vos documents, puis ouvrez-le avec pyzo.

Lorsqu'on joue au Scrabble, chaque joueur tire 7 lettres au hasard et doit trouver le mot qui permet d'obtenir le meilleur score en utilisant ces lettres.
Par exemple, avec les lettres `aeurtum` les mots possibles sont : **mure**, **muret**, **tueur** , **amer**...
Un joueur n'a pas l'obligation de former un mot de 7 lettres, le mot peut être plus court. La seule contrainte est que le mot ne peut être constitué que des lettres que le joueur a tiré.
Au Scrabble, chaque lettre est pondérée par un score qui dépend de la difficulté à utiliser cette lettre dans un mot.

Votre objectif est de trouver le mot qui marque le plus de points en utilisant les lettres disponibles (1 à 7 lettres).

Exercice 1 Occurences d'une lettre donnée dans un mot

1. `occurences(lettre, mot)` qui pour une lettre et un mot (chaînes de caractères) donnés, renvoie le nombre d'occurences de la lettre dans ce mot (la fonction doit renvoyer 0 si la lettre n'est pas présente).

```
def occurences(lettre, mot):
```

   Tester tous les cas.

2. **Comptage à l'aide d'un dictionnaire**.
   La fonction `Counter(sequence)` de la bibliothèque `collections` permet d'obtenir un dictionnaire contenant le nombre d'occurences de chaque élément de la séquence.

```
from collections import Counter
mot = "bonjour"

>>> Counter(mot)
Counter({'b': 1, 'o': 2, 'n': 1, 'j': 1, 'u': 1, 'r': 1})
```

   En déduire une autre façon d'obtenir `occurences(lettre, mot)` avec cette fonction.

```
def occurences2(lettre,mot):
    dict = Counter(mot)
    if lettre in dict :

        return ..................

    else :

        return .......
```

Le **Scrabble** se joue avec 100 jetons :
— 1 point : E ×15, A ×9, I ×8, N ×6, O ×6, R ×6, S ×6, T ×6, U ×6, L ×5
— 2 points : D ×3, M ×3, G ×2
— 3 points : B ×2, C ×2, P ×2
— 4 points : F ×2, H ×2, V ×2
— 8 points : J ×1, Q ×1
— 10 points : K ×1, W ×1, X ×1, Y ×1, Z ×1

Exercice 2 Score d'un mot donné

On donne maintenant un dictionnaire `scrabble` précisant le nombre de points pour chaque lettre au Scrabble.

```
scrabble = {"a":1,"b":3,"c":3,"d":2,"e":1,"f":4,"g":2,"h":4,"i":1,
"j":8,k":10,"l":1,"m":2,"n":1,"o":1,"p":3,"q":8,"r":1,"s":1,"t":1,
"u":1,"v":4,"w":10,"x":10,"y":10,"z":10}
```

On souhaite déterminer le nombre de points d'un mot donné.

Compléter la fonction `score_mot(mot)` qui prend en entrée un mot et qui donne en sortie le score du mot au Scrabble. Vous utiliserez la liste **scrabble** prédéfinie.

```
def score_mot(mot):
    '''score_mot(mot : str en minuscule) -> nombre de points'''
    pts = ...

    for ......... in mot:

        pts = pts + ..................

    return pts
```

Exercice 3 Piocher 7 lettres au hasard

On se donne un dictionnaire `jetons` donnant le nombre de jetons pour chaque lettre.

```
jetons = {"a":9,"b":2,"c":2,"d":3,"e":15,"f":2,"g":2,"h":2,"i":8,
"j":1,"k":1,"l":5,"m":3,"n":6,"o":6,"p":2,"q":1,"r":6,"s":6,"t":6,
"u":6,"v":2,"w":1,"x":1,"y":1,"z":1}
```

1. Que fait le script suivant ?

```
L_jetons = []
for lettre in jetons:
    L_jetons = L_jetons + [lettre] * jetons[lettre]

print(L_jetons)
```

2. Comment choisir au hasard un entier entre 0 et 99 ?

```
from .................. import ...........
```

3. Créer une fonction `pioche()` qui donne en sortie 7 lettres au hasard formant un dictionnaire.

```
>>> pioche()
{'e': 2, 'i': 1, 'l': 2, 'v': 2}
```

```
def pioche() :
    dict = {} # dictionnaire vide

    for k in range(....):

        numero = ..............  # entier aléa. entre 0 et 99

        lettre = L_jetons[.......] # lettre associée

        if lettre not in dict :

            dict[lettre] = 1

        else :

            dict[lettre] = ..............

    return dict
```

Exercice 4 **Meilleur mot avec une pioche donnée**

1. Écrire une fonction `possible(mot,lettres)` qui prend en entrée un mot (str) et un dictionnaire contenant les lettres disponibles avec leurs occurrences et renvoyant un booléen indiquant si le mot formé peut être composé avec les lettres fournies.
   Par exemple :

```
>>> lettres =  {'m': 2, 'n': 1, 'a': 2, 't':1}
>>> possible("maman",lettres)
True
>>> possible("baton",lettres)
False
```

```
def possible(mot,lettres):
    '''mot de type str, lettres de type dictionnaire
    contenant la lettre et son nombre d'occurrences'''
```

2. Le fichier `motsfrancais7.txt` contient une liste des mots français de 7 lettres ou moins. On va l'importer pour pouvoir l'utiliser.

```
import os
os.chdir('P:\Documents\')
fichier = open('motsfrancais7.txt','r', encoding='utf8')
dico = fichier.read().splitlines()  # liste des mots
fichier.close()
```

On dispose alors de la liste `dico` des mots français de moins de 7 lettres.

3. Écrire un programme qui pioche 7 lettres au hasard puis détermine le mot raportant de plus de points que l'on peut écrire avec ces lettres.

```
lettres = ..............  # lettres piochées
print(lettres)

scoremax = 0
motmax = ''

for mot in dico:

    if ..............................:

        score = ..................

        if .............. > scoremax:

            scoremax = ......................

            motmax = ..................

print(motmax, scoremax)
```